

Projet de programmation n°4

Achat d'ordinateurs

Léo Lellouche

```
MacBook-Pro-de-leo-2:ordi Leo$ ./a.out
```

```
Avez-vous besoin d'utiliser des logiciels spécifiques ?
```

```
A. Oui
```

```
B. Non
```

```
A
```

```
A. Door 7
```

```
B. Door 8
```

```
C. MacOS
```

```
D. PhotoStore
```

```
E. Scientifix
```

```
F. Penguin
```

```
B
```

```
Voulez-vous définir une quantité de RAM minimale ?
```

```
A. Oui
```

```
B. Non
```

```
A
```

```
Combien souhaitez-vous de RAM ?
```

```
2
```

```
Quel est votre budget maximal ?
```

```
550
```

```
Voulez-vous définir une consommation maximale ?
```

```
A. Oui
```

```
B. Non
```

```
A
```

```
Quel est la consommation maximale ?
```

```
220
```

```
Le PC pré-monté disponible dans le catalogue à 439 euros correspond a 80 % a vos besoins
```

```
Le PC pré-monté disponible dans le catalogue à 620 euros correspond a 60 % a vos besoins
```

```
Le PC pré-monté disponible dans le catalogue à 700 euros correspond a 60 % a vos besoins
```

I) Le projet

Nous devons ouvrir une base donnée disponible au format csv. Cette base de données contenant divers composants ainsi que leurs caractéristiques.

II) Cahier des charges :

Fonctionnalité de base :

- ✓ Faire différentes structures claires pour chaque composant
Chaque composant à une structure avec autant de variable que de spécificité du composant. Il est donc très facile de voir les caractéristiques de n'importe quel composant !
- ✓ Ouvrir et lire les fichiers quel que soit le nombre de lignes
- ✓ Extraire toutes les informations dans les structures préalablement créés
- ✓ Demander à l'utilisateur de saisir différentes caractéristiques
Ici, l'utilisateur peut choisir la RAM, les softs à utiliser, le prix et la consommation.

Fonctionnalité bonus :

- ✓ Affiche le taux de critère qui correspondent au choix de l'utilisateur

III) Bonus en dehors du cahier des charges

- ✓ Aucune fuite mémoire (leaks)
Quel que soit l'endroit dans le programme, toute mémoire non utilisée est proprement désallouée.
- ✓ Extrêmement facile d'ajouter d'autres critères.
Le programme chargeant l'intégralité de la base de donnée avant de sélectionner les critères, il suffit juste d'ajouter une fonction établissant le critères dans choix_utilisateur, et de modifier une constante dans Affichage_config_possible

IV) Algorithme

I) Structures

s_donnees : Structure contenant plusieurs tableaux de structures. Il y a un tableau pour les cartes mères, pour la RAM, le processeur...

s_carte_mere, s_RAM, s_processeur, s_graphic, s_HDD, s_ecran, s_soft, s_config : Structures qui contiennent les informations contenus dans les fichiers csv. Il y a autant de champ dans chaque structures qu'il y en a dans les fichiers csv.

s_besoin : Structures contenant les différents choix de composant de l'utilisateur

II) Main (main.c)

Variables locales :

base_donnees : Structure de type s_donnees, va stocker toutes la base de donnée

fd : tableau d'entiers, va stocker les file descriptor des fichiers csv

size : tableau d'entiers, va stocker le nombre de ligne de chaque fichier csv

DEBUT

```
| fd ← Réserver 8 entiers
| size ← Réserver 8 entiers
| ouverture_fichiers(fd, size) //Fonction qui va ouvrir les fichiers et récupérer le nombre de ligne de ceux-ci
| base_donnees = recup_base_donnees(fd, size)
| choix_utilisateur(base_donnees, size)
| fermeture_fichiers(fd, size) //Permet de fermer chaque fichier et de libérer la mémoire réservé par fd et size
```

FIN

III) Recup Base Donnees (main.c)

Paramètres :

fd : tableau d'entier contenant les files descriptor des fichiers ouvert

Size : tableau d'entier qui contient le nombre de ligne de chaque fichiers

Variable locale :

Donnees : Structure s_donnees, structure qui contient la base de donnée qui va être récupérée

Retour : donnees, la structure s_donnees contenant toute la base de donnée récupérée

DEBUT

```
|      Recup_cartes_meres(fd[0], size[0], donnees) //Envoie le file descriptor du premier fichier ouvertes ainsi que sa taille
|      Recup_processeurs(fd[1], size[1], donnees)
|      Recup_graphic(fd[2], size[2], donnees)
|      Recup_RAM(fd[3], size[3], donnees)
|      Recup_HDD(fd[4], size[4], donnees)
|      Recup_ecran(fd[5], size[5], donnees)
|      Recup_softs(fd[6], size[6], donnees)
|      Recup_config(fd[7], size[7], donnees)
```

FIN

IV) Recup Cartes meres (recup_base_donnees.c)

Paramètres :

fd : Files descriptor des fichiers cartes mères

Size : Nombre de ligne du fichier cartes mères

Variable locale :

Chaine lu : chaine de caractère contenant la ligne lu

Chaine transcrite : Tableau de chaine de caractères

i : Entier, compteur

DEBUT

```
|      Donnees->cartes_mere ← Réserver 'size' entiers
|      i ← 0
|      Tant que (get_next_line(fd, chaine_lu) ≠ 0) //La fonction Get_next_line lit une ligne dans un fichier et retourne 0 lorsqu'il n'y a rien à lire
|      |      chaine_transcrite = ft_strsplit(chaine_lu, ';') //La fonction strsplit va transformer une chaine de caractères en tableau 2D découpé selon le ';'
|      |      Chaine_lu ← libérer
|      |      chaine_lu ← NULL
|      |      données->cartes_mere[i].base.marque ← Réserver taille de chaine_transcrite[0]
|      |      données->cartes_mere[i].base.marque ← copie de chaine_transcrite[0]
|      |      données->cartes_mere[i].base.modele ← Réserver taille de chaine_transcrite[1]
|      |      données->cartes_mere[i].base.modele ← copie de chaine_transcrite[1]
|      |      données->cartes_mere[i].socket ← Réserver taille de chaine_transcrite[2]
|      |      données->cartes_mere[i].socket ← copie de chaine_transcrite[2]
|      |      données->cartes_mere[i].nb_socket ← atoi(chaine_transcrite[3]) //Atoi transforme une chaine de caractères en entier
|      |      // Ainsi de suite pour les 16 autres caractéristiques de la carte mère
|      |      ft_free_2d(chaine_transcrite, 20) //Libère le tableau 2D chaine_transcrite qui contient 20 lignes
|      |      i ← i + 1
|      FIN TANT QUE
|      chaine_lu ← Libérer
```

FIN

V) Choix Utilisateur (main.c)

Paramètres :

Base_donnees : structure s_donnees, base de donnée

size_struct : tableau d'entier contenant le nombre de cartes_mères, de RAM... différentes

Variable locale :

Besoin : structure s_besoin contenant les besoins de l'utilisateur

```
DEBUT
|   choix_softs(besoin, base_donnees, size_struct[6])
|   choix_RAM(besoin)
|   choix_prix(besoin)
|   choix_conso(besoin)
|   affichage_config_possible(base_donnees, besoin size_struct[7])
FIN
```

VI) Choix_softs (sélection ordi.c)

Paramètres :

Besoin : structure s_besoin, stockera les besoins de l'utilisateur

donnees : structure s_donnees, stock la base de donnée

size_struct_soft : Nombre de software dans la base de donnée

Variable locale :

Besoin : saisie, chaine de caractère, stock la saisie utilisateur

i : entier, compteur

```
DEBUT
|   Afficher « Avez-vous besoin d'utiliser des logiciels spécifiques ? »
|   saisie ← Saisir
|   Si (Saisie = Non)
|       |   besoin->perf_proco_min ← 0
|       |   besoin->RAM_min ← 0
|       |   besoin->graphic_min ← 0
|       |   Retourner
|   FIN SI
|   i ← 0
|   Afficher le nom de tous les logiciels
|   Si (
|       choix_conso(besoin)
|       besoin->perf_proco_min ← donnees.softs[saisie[0] - 'A'].processeur;
|       besoin->RAM_min ← donnees.softs[saisie[0] - 'A'].RAM
|       Si le logiciel sélectionné n'a pas de besoins graphiques (un X)
|           |   besoin->graphic_min ← 0
|       Sinon
|           |   besoin->graphic_min ← donnees.softs[saisie[0] - 'A'].graphic
|   FIN SI
FIN
```

VII) Affichage Config Possible (sélection ordi.c)

Paramètres :

Besoin : structure s_besoin, stockera les besoins de l'utilisateur

donnees : structure s_donnees, stock la base de donnée

nb_configs : Nombre de configurations existantes

Variable locale :

Critères_validé : entier, nombre de critère qui sont correcte

Critères_totaux : Constante, nombre de critère totaux

i : entier, compteur

conso_totale : entier, stock la consommation totale de la configuration

```
DEBUT
|   i ← 0
|   Tant Que (i < nb_configs)
|       |   criteres_validé ← 0
|       |   SI (donnees.configs[i].graphic->perf >= besoin.graphic_min)
|       |       |   criteres_validé ← criteres_validé + 1
|       |       |   SI (donnees.configs[i].processeur->perf >= besoin.perf_proco_min)
|       |       |       |   criteres_validé ← criteres_validé + 1
|       |       |       |   SI (donnees.configs[i].RAM->capacite >= besoin.RAM_min)
|       |       |       |       |   criteres_validé ← criteres_validé + 1
|       |       |       |       |   SI (donnees.configs[i].prix <= besoin.prix_max)
|       |       |       |       |       |   criteres_validé ← criteres_validé + 1
|       |       |   conso_totale ← donnees.configs[i].carte_mere->base.conso + donnees.configs[i].RAM->base.conso +
|       |       |                                   donnees.configs[i].processeur->base.conso +
|       |       |                                   donnees.configs[i].graphic->base.conso +
|       |       |                                   donnees.configs[i].HDD->base.conso
|       |       |   Si (conso_totale <= besoin.conso_max)
|       |       |       |   Criteres_validé ← Criteres_validé + 1
|       |       |   FIN SI
|       |       Afficher « Le PC suivant correspond à » '(criteres_validé / critère totaux) * 100' « a vos besoin »
|       |       i ← i + 1
|   FIN TANT QUE
FIN
```